파이썬 EDA

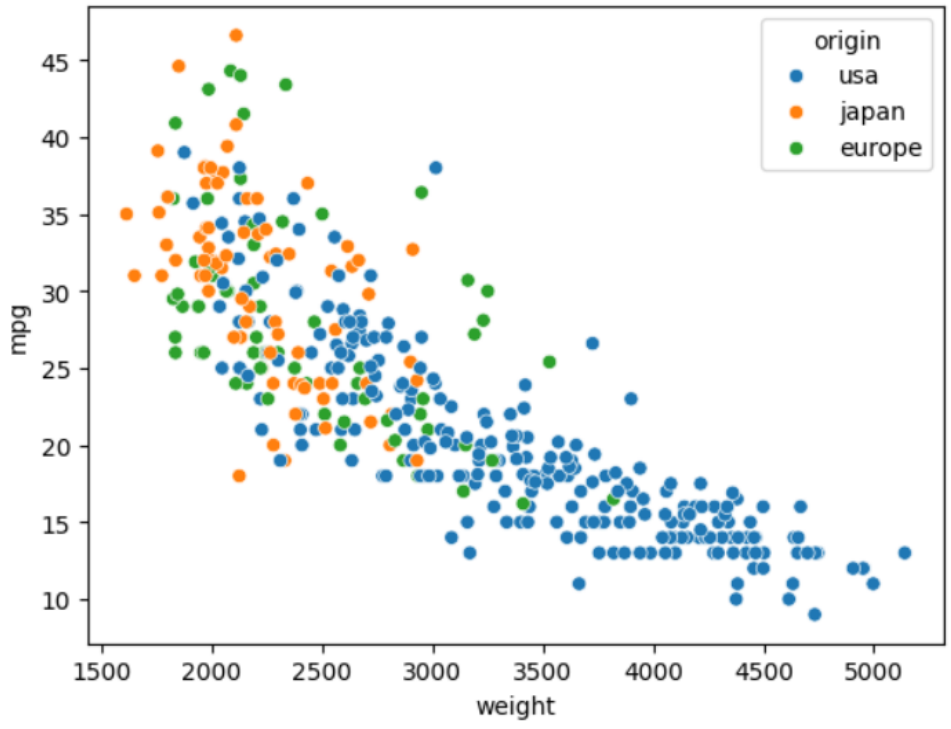
생성자	선 선우 박
생성 일시	@2024년 8월 19일 오후 2:08
태그	인프런 파이썬

두 개 이상의 수치형 변수 시각화

- 2개 이상의 수치변수 비교하기
  - scatterplot
  - 양의 상관이 있는지 음의 상관이 있는지 알아볼 수 있음
  - hue로 색상 표현(origin : origin 값에 따라 색상 다르게)

```
1 # scatterplot 을 통해 2개의 수치변수 비교하기
2 sns.scatterplot(data=df, x='weight', y='mpg', hue = "origin")
```

```
<Axes: xlabel='weight', ylabel='mpg'>
```

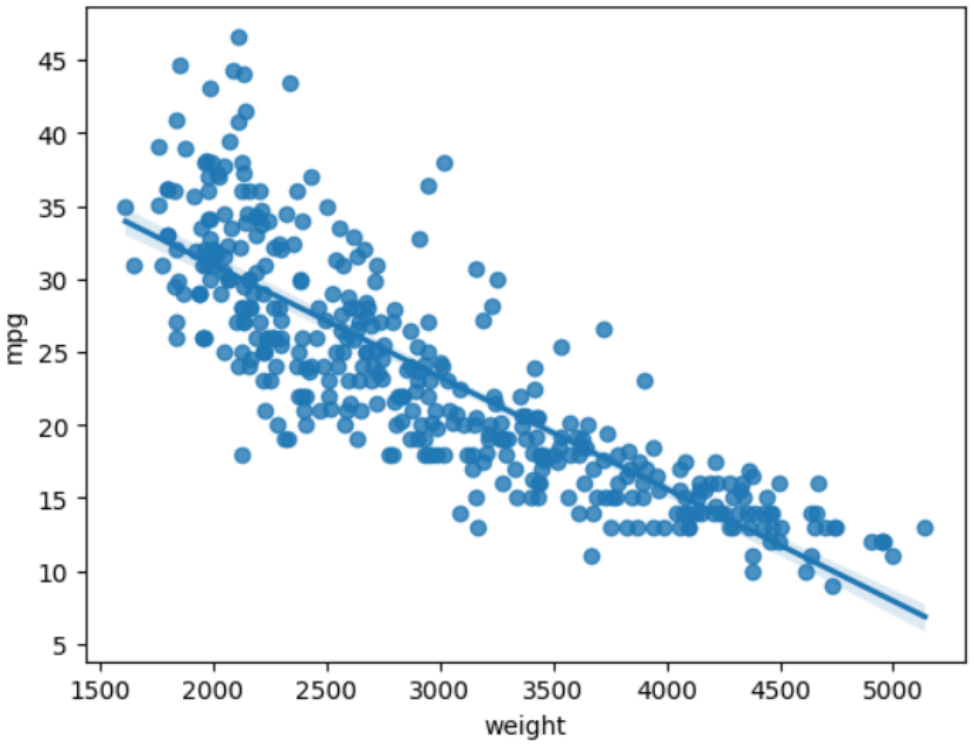


- 회귀 시각화

- regplot

```
1 #regplot 으로 회귀선 그리기
2 # 회귀선을 그려서 weight 값이 증가하면 연비가 줄어든다는 것을 확인할 수 있음
3 sns.regplot(data=df, x='weight', y='mpg')
```

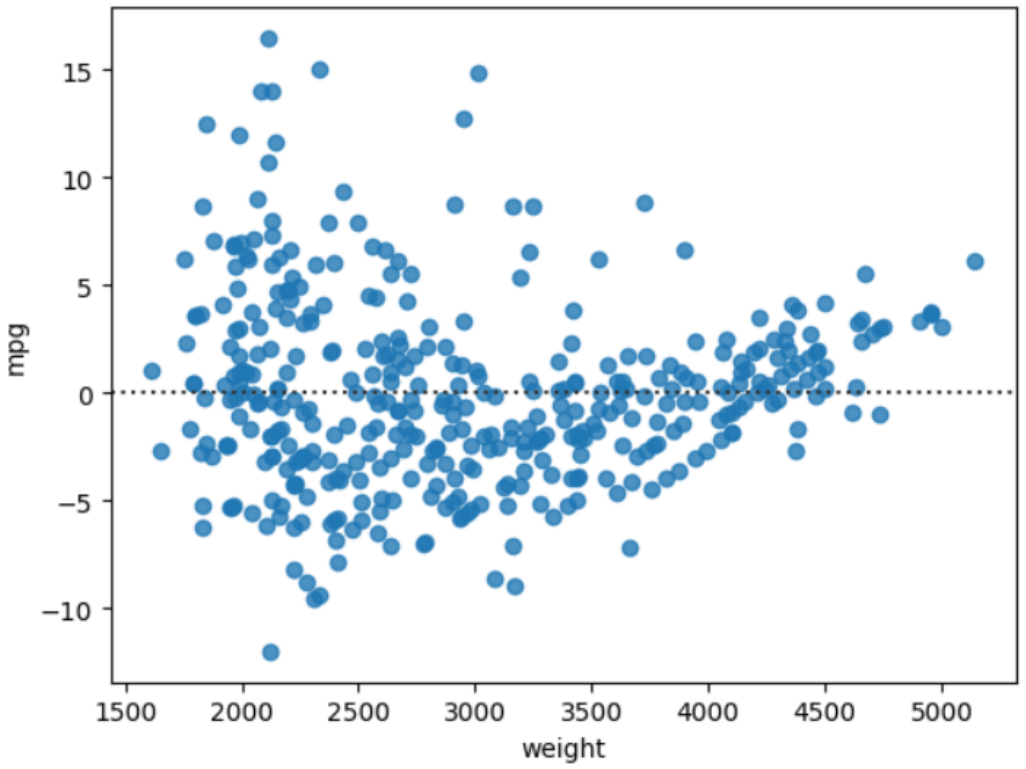
```
<Axes: xlabel='weight', ylabel='mpg'>
```



- 잔차 시각화
  - residplot
  - lmplot
  - jointplot

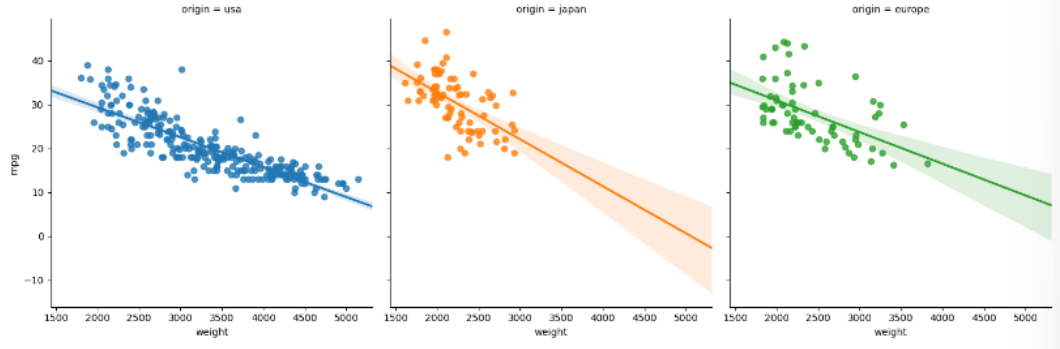
```
# 회귀선의 잔차를 시각화 하기
# 기울어진 회귀선을 수평으로
# 회귀선에 가까운 값들을 0으로 표시, 차이가 나는 값들이 얼마나 떨어져 있는지 표시
sns.residplot(data=df, x='weight', y='mpg')
```

```
<Axes: xlabel='weight', ylabel='mpg'>
```

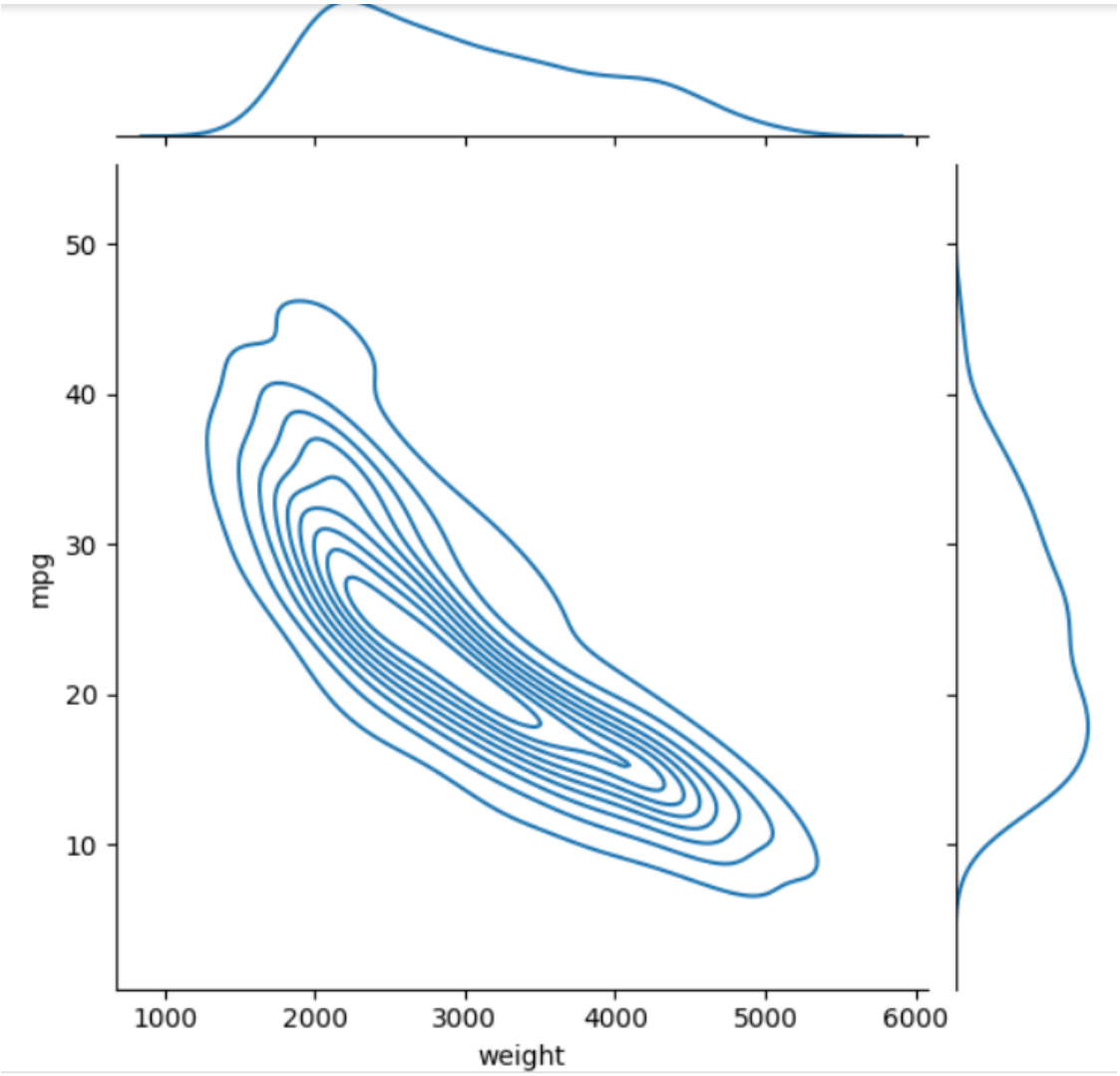



```
# lmplot 을 통해 범주값에 따라 색상, 서브플롯 그리기
# col 옵션을 이용해서 서브플롯 그릴 수 있음
# truncate=False -> 회귀선을 자르지 않고 보여줌
sns.lmplot(data=df, x="weight", y="mpg", hue="origin", col="origin", truncate=False)
```

```
<seaborn.axisgrid.FacetGrid at 0x7a09e5908eb0>
```




```
# jointplot 2개의 수치변수 표현하기
# kind옵션 : kde(밀도추정), reg(회귀선), hex(히스토그램과 스캐터플롯)
sns.jointplot(data=df, x="weight", y="mpg", kind="kde")
```



- pariplot

```
1 # pairplot 은 시간이 오래 걸리기 때문에 일부 샘플을 추출해 그려보고(100개)
2 # 샘플의 수를 늘려가며 그리는 걸 추천합니다.
3 # df_sample
4 df_sample = df.sample(100)
5 df_sample.shape
```

```
(100, 9)
```

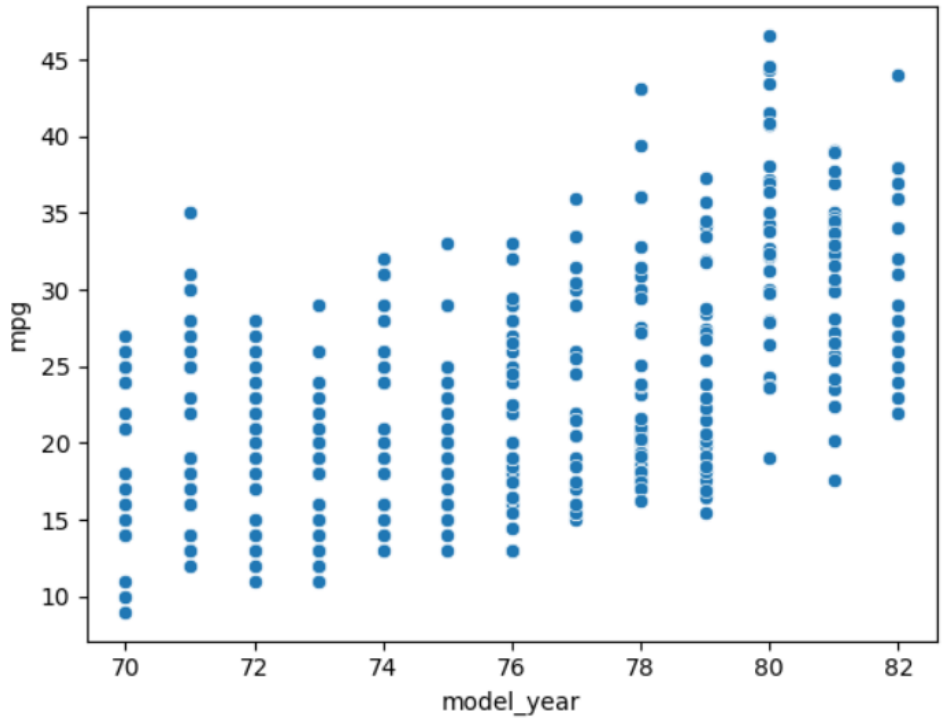
```
1 # origin 값에 따라 다른 색상으로 그리기
2 # 모든 수치 변수에 대해 짝을 지어서 그려줌
3 sns.pairplot(df_sample, hue="origin")
```

```
<seaborn.axisgrid.PairGrid at 0x7a09e3607940>
```

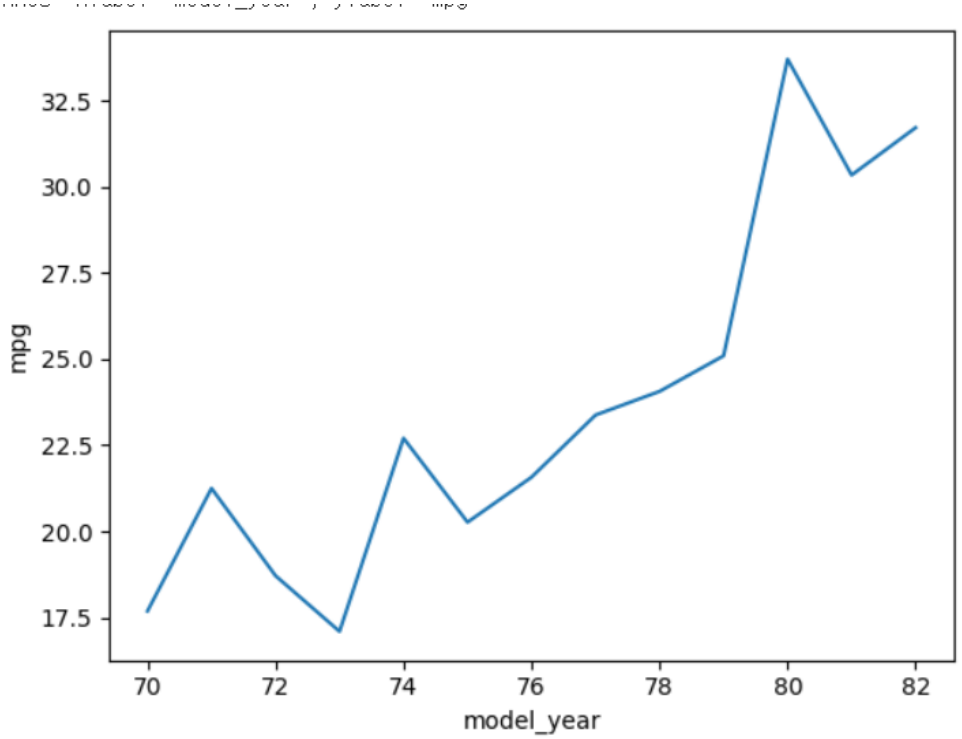
- lineplot

```
1 sns.scatterplot(data=df, x='model_year', y='mpg')
```

```
<Axes: xlabel='model_year', ylabel='mpg'>
```

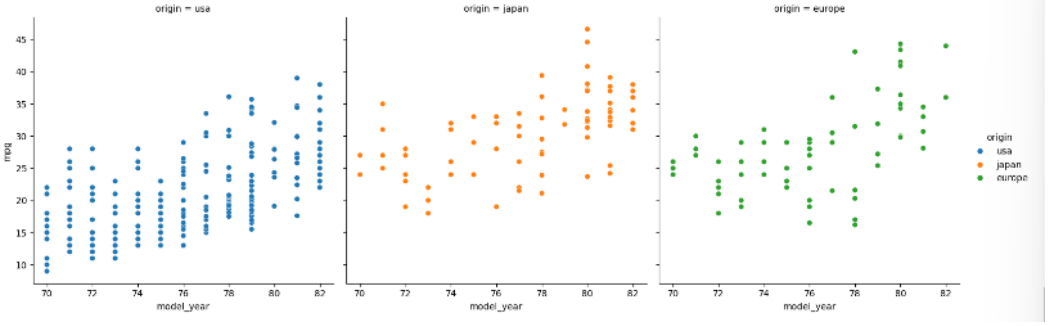



```
1 # lineplot으로 model_year, mpg를 시각화 합니다.
2 # model_year에 따라서 mpg 값이 어떻게 변화하는지
3 # ci=None -> 신뢰구간 생략
4 sns.lineplot(data=df, x='model_year', y='mpg', ci=None)
```



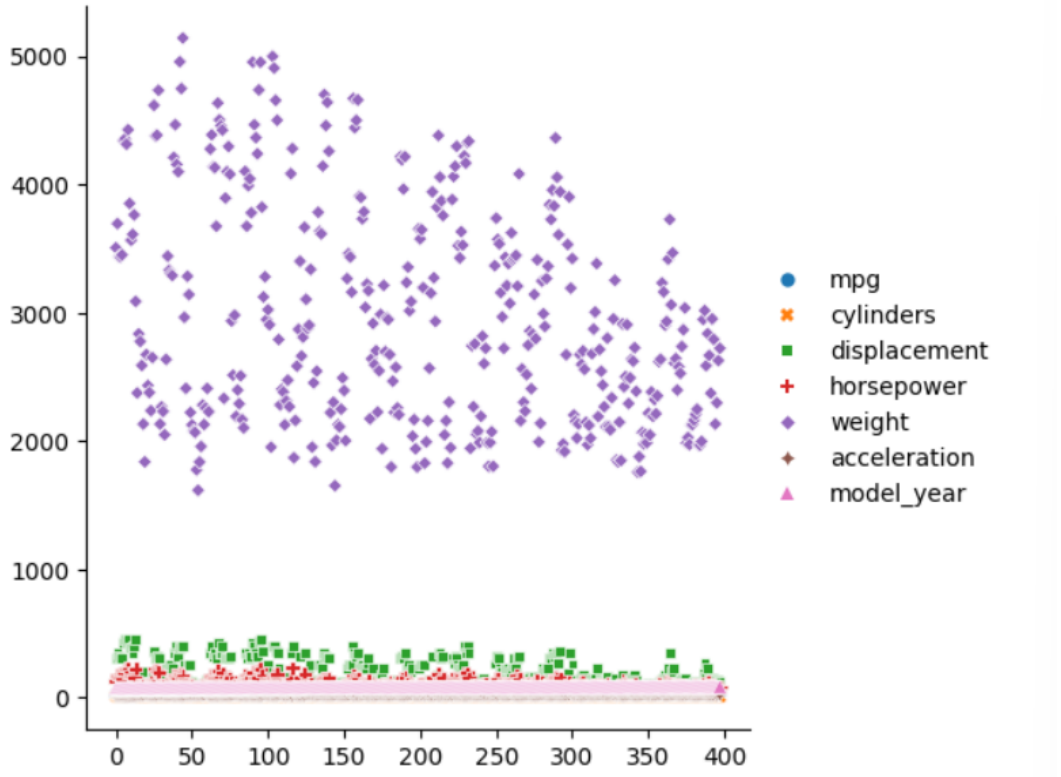

```
1 # relplot 으로 수치 변수에 따라 서브플롯을 그립니다.
2 # 수치형 변수에 대한 관계 표현
3 sns.relplot(data=df, x="model_year", y="mpg", hue="origin", col="origin")
```

```
<seaborn.axisgrid.FacetGrid at 0x7a09dece5840>
```

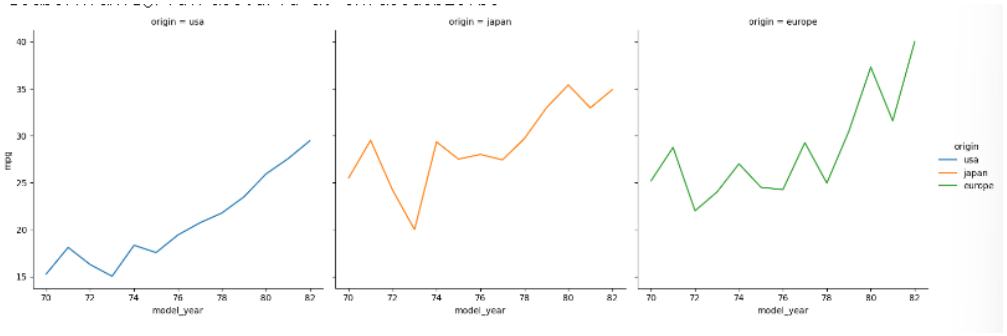


```
1 # relplot 으로 전체 수치 변수에 대한 시각화를 합니다.
2 sns.relplot(data=df) # 스케일 값이 달라서 보기 좋지 않음
```

```
<seaborn.axisgrid.FacetGrid at 0x7a09dee2eb00>
```




```
1 # relplot 의  kind 옵션을 통해 선그래프를 그립니다.
2 sns.relplot(data=df, x="model_year", y="mpg", hue="origin", col="origin",
3               kind="line", ci=None)
```

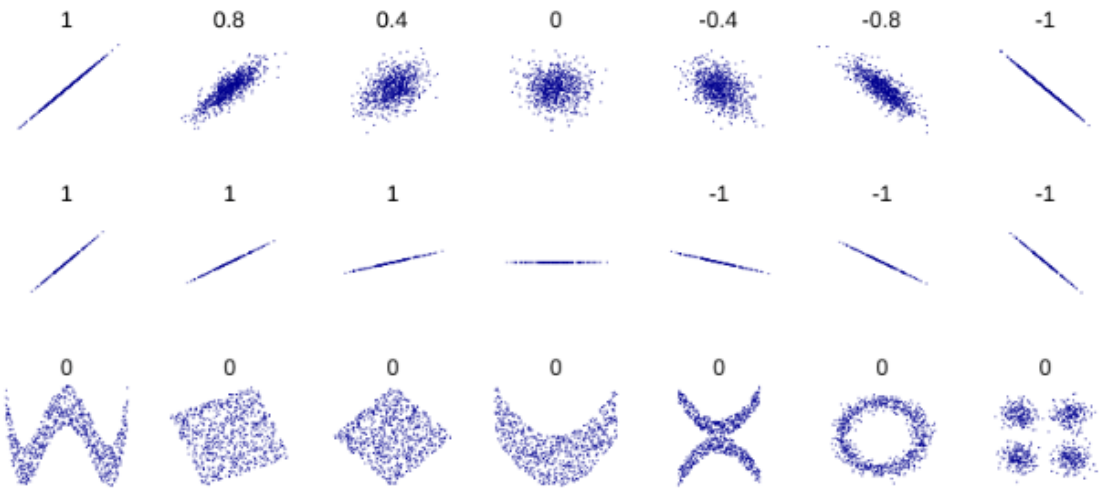


상관분석

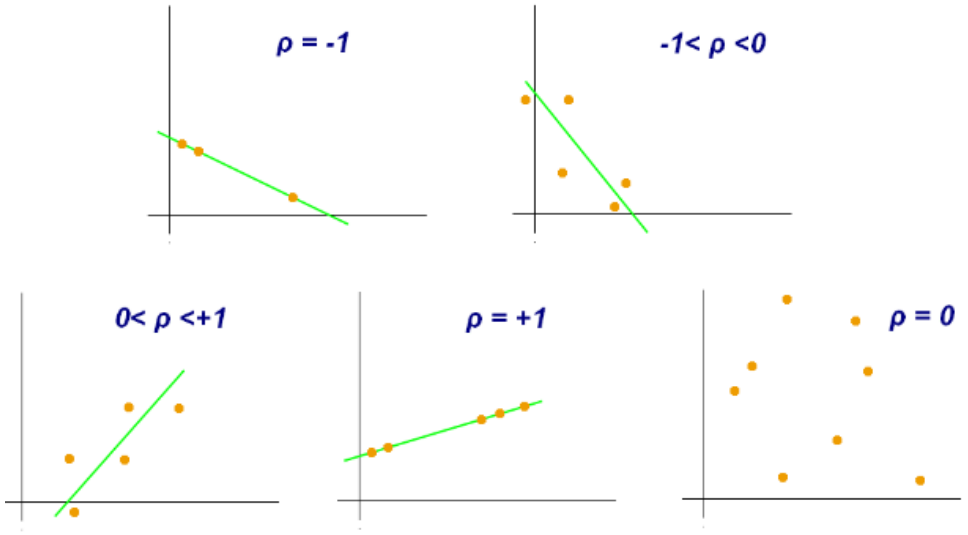
- 상관 분석(相關 分析, Correlation analysis) 또는 '상관관계' 또는 '상관'은 확률론과 통계학에서 두 변수간에 어떤 선형적 또는 비선형적 관계를 갖고 있는지를 분석하는 방법이다. 두 변수는 서로 독립적인 관계이거나 상관된 관계일 수 있으며 이때 두 변수간의

관계의 강도를 상관관계(Correlation, Correlation coefficient)라 한다. 상관분석에서는 상관관계의 정도를 나타내는 단위로 모상관계수로 ρ를 사용하며 표본 상관 계수로 r 을 사용한다.

- 상관관계의 정도를 파악하는 상관 계수(相關係數, Correlation coefficient)는 두 변수간의 연관된 정도를 나타낼 뿐 인과관계를 설명하는 것은 아니다. 두 변수간에 원인과 결과의 인과관계가 있는지에 대한 것은 회귀분석을 통해 인과관계의 방향, 정도와 수학적 모델을 확인해 볼 수 있다.
- **피어슨 상관계수**
  - r 값은 X 와 Y 가 완전히 동일하면 +1, 전혀 다르면 0, 반대방향으로 완전히 동일 하면 –1 을 가진다.
  - 결정계수(coefficient of determination)는 r^2 로 계산하며 이것은 X 로부터 Y 를 예측할 수 있는 정도를 의미한다.



```
일반적으로
r이 -1.0과 -0.7 사이이면, 강한 음적 선형관계,
r이 -0.7과 -0.3 사이이면, 뚜렷한 음적 선형관계,
r이 -0.3과 -0.1 사이이면, 약한 음적 선형관계,
r이 -0.1과 +0.1 사이이면, 거의 무시될 수 있는 선형관계,
r이 +0.1과 +0.3 사이이면, 약한 양적 선형관계,
r이 +0.3과 +0.7 사이이면, 뚜렷한 양적 선형관계,
r이 +0.7과 +1.0 사이이면, 강한 양적 선형관계
```



- 상관계수 구하기

```python
# 데이터프레임 전체의 수치변수에 대해 상관계수를 구합니다.
# 피어슨상관계수가 디폴트
corr = df.select_dtypes(include='number').corr()
corr
```

	mpg	cylinders	displacement	horsepower	weight	acceleratior
mpg	1.000000	-0.775396	-0.804203	-0.778427	-0.831741	0.420289
cylinders	-0.775396	1.000000	0.950721	0.842983	0.896017	-0.505419
displacement	-0.804203	0.950721	1.000000	0.897257	0.932824	-0.543684
horsepower	-0.778427	0.842983	0.897257	1.000000	0.864538	-0.689196
weight	-0.831741	0.896017	0.932824	0.864538	1.000000	-0.417457
acceleration	0.420289	-0.505419	-0.543684	-0.689196	-0.417457	1.000000
model_year	0.579267	-0.348746	-0.370164	-0.416361	-0.306564	0.288137

```
# np.triu : matrix를 상삼각행렬로 만드는 numpy math
# [1 2 3]   np.triu  [1 2 3]
# [4 5 6]   -------> [0 5 6]
# [2 3 4]            [0 0 4]
# np.ones_like(x) : x와 크기만 같은 1로 이루어진 array를 생성

# 수식적으로 어려워 보일수도 있지만 간단함
# 자기상관계수는 대각행렬을 기준으로 대칭되어 같은 값이 출력되므로,
# 이대로 전체를 heatmap을 plot하면 오히려 가독성이 떨어질 수 있음
# 이에, 가독성을 높이기 위해 대각행렬 기준으로
# 한쪽의 데이터들만 masking 기법을 통해 plot하여
# 가독성을 높이는 효과를 가질수 있음
# np.ones_like로 heatmap의 마스크값 구하기
# np.ones_like(corr) -> 1로 채워진 행렬
# mask -> 상삼각행렬
mask = np.triu(np.ones_like(corr))
mask
```

```
array([[1., 1., 1., 1., 1., 1., 1.],
       [0., 1., 1., 1., 1., 1., 1.],
       [0., 0., 1., 1., 1., 1., 1.],
       [0., 0., 0., 1., 1., 1., 1.],
       [0., 0., 0., 0., 1., 1., 1.],
       [0., 0., 0., 0., 0., 1., 1.],
       [0., 0., 0., 0., 0., 0., 1.]])
```

```
print(plt.colormaps())
```

```
['magma', 'inferno', 'plasma', 'viridis', 'cividis', 'twilight', 'twilight_shifted', 'turb
```

```python
# heatmap 을 통해 상관계수를 시각화 합니다.
# cmap옵션 : coolwarm-빨간색,파란색으로 / Blues -> 한가지색으로 RdBu_r-> -값일때 파란
# annot=True -> 상관계수 표현
# mask=mask -> 대각선 위쪽의 값 날
sns.heatmap(corr, annot=True, cmap="RdBu_r", vmax=1, vmin=-1, mask=mask)
```

```
<Axes: >
```